Listing of Claims

1   Claim 1 (Currently Amended): A method of implementing an atomic transaction using
2   a program logic, said method comprising: requesting in said program logic a transaction
3   identifier for said atomic transaction; generating said transaction identifier in a transaction
4   manager in response to said requesting; specifying in said program logic a plurality of
5   combinations for execution in a sequential order, wherein each of said plurality of
6   combinations contains said transaction identifier, a task procedure, and a ~~roll-back~~ rollback
7   procedure, wherein said task procedure implements a part of said atomic transaction and said
8   ~~roll-back~~ rollback procedure is designed to ~~roll-back~~ rollback said task procedure; executing
9   said task procedures in said sequential order; keeping track of said ~~roll-back~~ rollback
10   procedures in said transaction manager; and executing said ~~roll-back~~ rollback procedures in
11   a reverse order of said sequential order if said atomic transaction is to be aborted, wherein
12   said ~~roll-back~~ rollback procedures are identified according to said keeping.

1   Claim 2 (Original): The method of claim 1, wherein said transaction identifier is
2   unique to each of the atomic transactions.

1   Claim 3 (Currently Amended): The method of claim 1, wherein said keeping
2   comprises storing data representing said ~~roll-back~~ rollback procedures in a stack.

1   Claim 4 (Original): The method of claim 3, wherein said stack is stored in a memory.

1   Claim 5 (Original): The method of claim 1, further comprising examining a status
2   returned by execution of one of said task procedures and performing said aborting if said
3   status indicates an error.

1   Claim 6 (Original): The method of claim 1, wherein said aborting is performed
2   asynchronously.

1   Claim 7 (Currently Amended): A computer readable medium carrying one or more
2   sequences of instructions representing a program logic for execution on a system, said

3   program logic implementing an atomic transaction, wherein execution of said one or more

4   sequences of instructions by one or more processors contained in said system causes said one

5   or more processors to perform the actions of: requesting an identifier for said atomic

6   transaction; setting a variable to equal said identifier; specifying a plurality of combinations

7   for execution, wherein each of said plurality of combinations contains said transaction

8   identifier, a task procedure, and a ~~roll-back~~ rollback procedure, wherein said task procedure

9   implements a part of said atomic transaction and said ~~roll-back~~ rollback procedure is designed

10   to ~~roll-back~~ rollback said task procedure; and aborting said atomic transaction by specifying

11   said identifier associated with an abort procedure to cause said ~~roll-back~~ rollback procedures

12   to be executed.

1   Claim 8 (Original): The computer readable medium of claim 7, wherein said

2   specifying comprises including each of said plurality of combinations in a single procedure

3   call.

1   Claim 9 (Original): The computer readable medium of claim 7, further comprising

2   examining a status returned by execution of one of said task procedures and performing said

3   aborting if said status indicates an error.

1   Claim 10 (Currently Amended): A computer readable medium carrying one or more

2   sequences of instructions for supporting implementation of an atomic transaction in a system,

3   wherein execution of said one or more sequences of instructions by one or more processors

4   contained in said system causes said one or more processors to perform the actions of:

5   generating an identifier for said atomic transaction; receiving a plurality of combinations for

6   execution, wherein each of said plurality of combinations contains said transaction identifier,

7   a task procedure, and a ~~roll-back~~ rollback procedure, wherein said task procedure implements

8   a part of said atomic transaction and said ~~roll-back~~ rollback procedure is designed to ~~roll-back~~

9   rollback said task procedure; executing said task procedures; and executing said ~~roll-back~~

10   rollback procedures in response to receiving an abort request.

1   Claim 11 (Currently Amended): The computer readable medium of claim 10, wherein

2   said task procedures are executed in an execution order and corresponding ~~roll-back~~ <u>rollback</u>

3   procedures are executed in a reverse order of said execution order.

1   Claim 12 (Currently Amended): The computer readable medium of claim 11, further

2   comprising storing data indicating that said ~~roll-back~~ <u>rollback</u> procedures are to be executed

3   in said reverse order to abort said atomic transaction.

1   Claim 13 (Currently Amended): The computer readable medium of claim 12, wherein

2   said <u>transaction</u> identifier is generated to be unique for each atomic transaction.

1   Claim 14 (Original): The computer readable medium of claim 12, wherein said data

2   is represented in the form of a stack.

3   Claim 15 (Original): The computer readable medium of claim 14, wherein said stack

4   is stored in a memory.

1   Claim 16 (Currently Amended): A computer system comprising: a memory storing

2   a plurality of instructions; and a processing unit coupled to said memory and executing said

3   plurality of instructions to support implementation of an atomic transaction in a programming

4   environment, said processing unit being operable to: request in a program logic a transaction

5   identifier for said atomic transaction; generate said transaction identifier in a transaction

6   manager in response to said requesting; specify in said program logic a plurality of

7   combinations for execution in a sequential order, wherein each of said plurality of

8   combinations contains said transaction identifier, a task procedure, and a ~~roll-back~~ <u>rollback</u>

9   procedure, wherein said task procedure implements a part of said atomic transaction and said

10   ~~roll-back~~ <u>rollback</u> procedure is designed to ~~roll-back~~ <u>rollback</u> said task procedure; execute

11   said task procedures in said sequential order; keep track of said ~~roll-back~~ <u>rollback</u> procedures

12   in said transaction manager; and execute said ~~roll-back~~ <u>rollback</u> procedures in a reverse order

13   of said sequential order if said atomic transaction is to be aborted, wherein said ~~roll-back~~

14   <u>rollback</u> procedures are identified according to said keeping.

1   Claim 17 (Original): The computer system of claim 16, wherein said transaction

2   identifier is unique to each of the atomic transactions.

1   Claim 18 (Currently Amended): The computer system of claim 16, wherein said

2   processing unit is operable to store data representing said ~~roll-back~~ <u>rollback</u> procedures in a

3   stack to perform said keep.

1   Claim 19 (Original): The computer system of claim 18, wherein said stack is stored

2   in a memory.

1   Claim 20 (Original): The computer system of claim 16, wherein said processing unit

2   is further operable to examine a status returned by execution of one of said task procedures

3   and to perform said aborting if said status indicates an error.

1   Claim 21 (Currently Amended): The computer system of claim 16, wherein said

2   processing unit is operable to execute said ~~roll-back~~ <u>rollback</u> procedures asynchronously.